Riassunto di Sistemi in Tempo Reale LS

Silvia Cereda

July 10, 2007

1 Schedulazione di processi periodici

Condizione necessaria (ma non sufficiente) affinché un insieme di N processi sia schedulabile è che il fattore di utilizzazione del processore non superi 1:

$$U = \sum_{j=1}^{N} \frac{C_j}{T_j} \leq 1 \tag{1}$$

con C_j il tempo massimo di esecuzione e T_j il periodo di esecuzione del processo j-esimo.

L'equazione 1 è condizione necessaria e sufficiente per la schedulazione per processi con periodi in relazione armonica.

1.1 Schedulazione clock-driven

Schedulazione di tipo off-line, non-preemptive e garantita (se esiste un feasible schedule). Tutti i vincoli sono soddisfatti a priori durante la costruzione del feasible schedule.

1.1.1 Costruzione di un feasible schedule

1. Dimensionamento del ciclo maggiore

$$M = mcm\{T_j\} \tag{2}$$

2. Dimensionamento del ciclo minore

$$C_i \leq m \leq T_i, \forall i \tag{3}$$
$$Mmod(m) = 0 \tag{4}$$
$$2m - mcd(m, T_i) \leq T_i, \forall i \tag{5}$$

Se non esiste un feasible schedule da questa soluzione, si partiziona il/i processo/i con maggiore tempo di esecuzione in sottoprocessi; normalmente ordinando i due processi, quello con capacità minore viene ordinato più in basso (ovvero è meno prioritario). La partizione può essere necessaria più volte, nel caso non si trovi un feasible schedule ammissibile.

$$P_i(C_i.T_i) \Rightarrow P'_i(C'_i, T_i), P''_i(C''_i, T_i) \tag{6}$$

con $C'_i + C''_i = C_i$ e $P'_i \prec P''_i$. Nelle iterazioni non considero più i valori che sono già stati calcolati come non ammissibili per qualunque dei vincoli sopra citati.

Una volta dimensionato correttamente il ciclo minore, calcolo il numero dei cicli minori e dei job per ogni processo nell'ambito di ciascun ciclo maggiore

$$n_{cm} = \frac{M}{m} \qquad\qquad n_{J_i} = \frac{M}{T_i} \tag{7}$$

Proseguo poi a creare la tabella che avrà tante righe quanto il valore di n_{cm} e come colonne il numero di job per ogni processo, calcolati attraverso l'n_{J_i}.

Per compilarla devo inserire delle X nelle celle corrispondenti a ciascun ciclo minore per job in cui potrà essere eseguito il processo, ovvero all'interno di ogni periodo. Se il periodo non è multiplo del ciclo minore, devo calcolare l'intervallo entro cui il job sarà eseguito e posizionare le X solamente nelle celle in cui il job può essere sicuramente eseguito, evitando quindi quelle in cui le celle che comprendono istanti esterni al periodo corretot di esecuzione del job. Per esempio, con cicli minori di 2 unità di tempo e un processo con periodicità 5, non inserirò la X nella cella 5-6 perché il job_i non può essere eseguito oltre l'istante 5 e il job_{i+1} non può essere eseguito prima dell'istante 6.

Infine devo associare ogni processo al ciclo minore compilando l'ultima tabella secondo questi criteri di priorità per l'assegnamento:

1. Precedenza al processo con frequenza di esecuzione più elevata (T_i minore).
2. Precedenza al processo con tempo di esecuzione più elevato (C_i maggiore).
3. Precedenza al ciclo con minore tempo residuo libero in caso di associazioni alternative.

1.2 Schedulazione priority-driven

Ad ogni processo è associata una priorità e un'informazione che ne indica lo stato di esecuzione (idle, ready, running). Un processo in esecuzione è sospeso se un processo di priorità maggiore è pronto per l'esecuzione.

1.2.1 RMPO

L'algoritmo **Rate Monotonic Priority Ordering** associa staticamente priorità maggiore ai processi più frequenti $p(P_j) = p_j \propto \frac{1}{T_j} = \frac{1}{D_j}$.

Se un insieme di processi periodici è schedulabile on qualche algoritmo che prevede un'attribuzione statica di priorità, allora è schedulabile anche con RMPO.
L'analisi della schedulabilità avviene mediante la costruzione di diagrammi temporali, l'applicazione di criteri basati sul fattore di utilizzazione del processore, e mediante il calcolo dei tempi di riposta dei processi.

Diagrammi temporali

Vale l'equazione 1 (condizione necessaria) che è anche condizione sufficiente per i processi con periodi in relazione armonica.

Fattore di utilizzabilità del processore

Teorema di Liu-Layland: Condizione sufficiente (non necessaria) affinché un insieme di N processi sia schedulabile con RMPO è che valga:

$$U \leq U_{RMPO}(N) = N \cdot (2^{1/N} - 1) \tag{8}$$

Attenzione che se i teorema non è soddisfatto, non è detto che l'algoritmo non sia schedulabile con RMPO perché la condizione è solo sufficiente, ma non necessaria.

Corollario del teorema di Liu-Layland: Esplicitando l'"utilizzazione schedulabile dell'algoritmo RMPO" in funzione dei singoli fattori di utilizzazione dei processi si previene ad un test di schedulabilità meno conservativo

$$U_{RMPO}(U_1, \cdots, U_N) = \prod_{j=1}^{N}(1 + U_j) \leq 2 \tag{9}$$

Test di Kuo-Mok: Condizione sufficiente per cui un insieme di processi sia schedulabile con RMPO è che un insieme di sottoinsiemi disgiunti di processi in relazione armonica soddisfi l'equazione 8. Quindi unisco i processi che sono in relazione armonica tra loro, ricalcolo C e T e sulla base del nuovo schema applico il teorema di Liu-Layland.

Test di Burchard: Condizione sufficiente per cui un insieme di processi sia schedulabile con RMPO è che soddisfi l'equazione

$$U \leq U_{RMPO}(N, \zeta) \tag{10}$$

$$X_j = \log_2 T_j - \lfloor \log_2 T_j \rfloor, \forall j \tag{11}$$

$$\zeta = \max_{1 \leq j \leq N} X_j - \min_{1 \leq j \leq N} X_j \tag{12}$$

$$U_{RMPO}(N, \zeta) = \begin{cases} (N-1)(2^{\frac{\zeta}{N-1}} - 1) + 2^{1-\zeta} - 1 & \zeta < 1 - \frac{1}{N} \\ N(2^{\frac{1}{N}} - 1) & \zeta \geq 1 - \frac{1}{N} \end{cases} \tag{13}$$

Criterio di Han: Condizione sufficiente affinché un insieme di processi sia schedulabile con RMPO è che valga l'equazione di utilizzazione del processore (1) su un "insieme accelerato" di processi in relazione armonica con $C'_i = C_i$ e $T'_i \leq T_i \forall i$.

Calcolo dei tempi di risposta (Algoritmo di Audsley)

La schedulabilità è garantita (condizione necessaria e sufficiente) se per ogni processo i, il suo tempo di risposta non eccede la deadline, ovvero $R_i \leq T_i$ posto che:

$$R_i^0 = C_i \tag{14}$$

$$R_i^n = C_i + \sum_{j | p_j > pi} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{15}$$

Si evince che il processo a massima priorità non è mai soggetto a interferenza.

1.2.2 DMPO

L'algoritmo **Deadline Monotonic Priority Ordering** è una estensione del precedente modello per la schedulazione di processi sia periodici che sporadici e prevede che ogni processo P_j sia caratterizzato da 3 parametri: il periodo T_j, la deadline $D_j \leq T_j$ e la capacità $C_j < D_j$. L'algoritmo associa una priorità statica ad ogni processo inversamente proporzionale alla deadline di ogni processo $p_j \propto \dfrac{1}{D_j}$.

Diagrammi temporali

Vale l'equazione 1 (condizione necessaria) che è anche condizione sufficiente per i processi con periodi in relazione armonica.

Calcolo dei tempi di risposta (Algoritmo di Audsley)

La schedulabilità è garantita (condizione necessaria e sufficiente) se per ogni processo i, il suo tempo di risposta non eccede la deadline, ovvero $R_i \leq D_i$ posto che:

$$R_i^0 = C_i \tag{16}$$

$$R_i^n = C_i + \sum_{j|p_j>pi} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{17}$$

Si evince che il processo a massima priorità non è mai soggetto a interferenza.

Fattore di utilizzabilità del processore

- Condizione sufficiente (ma non necessaria)

$$C_i + \sum_{j|p_j>pi} \left\lceil \frac{D_i}{T_j} \right\rceil C_j \leq D_j \tag{18}$$

- Condizione sufficiente (ma non necessaria)

$$\Delta = \sum_{j=1}^{N} \frac{C_j}{D_j} \leq U_{RMPO}(N) = N(2^{1/N} - 1) \tag{19}$$

- **Test di Lehoczky**: condizione sufficiente (ma non necessaria)

$$\sum_{j=1}^{N} \frac{C_j}{T_j} \leq U(N,\delta) = \begin{cases} N((2\delta)^{(1/N)} - 1) + 1 - \delta & 0.5 \leq \delta \leq 1 \\ \delta & 0 \leq \delta \leq 0.5 \end{cases} \quad \delta = \min_j\{\delta_j\} \tag{20}$$

  in cui $\delta = \frac{Dj}{Tj}$.

- **Test di utilizzazione efficace dei processi:** si calcola il fattore di utilizzazione efficace f_j per ogni processo p_j

$$f_j = \sum_{i \in H_n} \frac{C_i}{T_i} + \frac{1}{T_j} \cdot \left(C_j + \sum_{k \in H_1} C_k \right) \tag{21}$$

in cui H_1 e H_n sono i due sottoinsiemi di processi di priorità maggiore o uguale a $p(P_j)$ con periodo rispettivamente maggiore o uguale a D_j o minore di D_j, poi si verifica se vale:

$$f_j \leq U(N = |H_n| + 1, \delta = \delta_j) \tag{22}$$

In presenza di Interrupt Service Routines (ISR) la schedulabilità è calcolabile mediante il test di utilizzazione efficace tenendo conto che le ISR vengono eseguite ad una priorità più elevata rispetto a quella normalmente assegnata.

1.2.3 EDF

L'algoritmo **Earliest Deadline First** assegna una priorità maggiore ai processi la cui deadline è più imminente.

- Condizione necessaria e sufficiente affinché un insieme di processi *periodici* sia schedulabile con EDF è che valga:

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq U_{EDF} = 1 \tag{23}$$

- Condizione sufficiente (ma non necessaria) affinché un insieme di processi *periodici e sporadici* sia schedulabile con EDF è che valga:

$$\Delta = \sum_{i=1}^{N} \frac{C_i}{D_i} \leq 1 \tag{24}$$

2 Schedulazione di processi aperiodici

Le richieste periodiche sono caratterizzate da vincoli temporali soft o non realtime e sono servite solo se non ci sono processi pronti per l'esecuzione.

2.1 Servizio in background

Le richieste vengono eseguite da un servizio che può essere considerato come un processo non periodico a priorità minore agli altri processi.

2.2 Servizio tramite server a priorità statica

2.2.1 Polling Server

Il server è un processo periodico con periodo T_s e tempo massimo di esecuzione C_s prefissati; ha una priorità assegnata con la stessa strategia dei processi periodici, tipicamente RMPO. La capacità è consumata durante il servizio di richieste aperiodiche oppure scartata se inutilizzata e viene ripristinata all'inizio di ogni periodo T_s.
La schedulabilità è garantita se vale:

$$U_p + U_s \leq (N+1)(2^{1/(N+1)} - 1) \tag{25}$$

2.2.2 Deferrable Server

A differenza del Polling Server, la capacità viene conservata in assenza di richieste aperiodiche pendenti.
Quando DS ha capacità positiva, anche se non ci sono richieste aperiodiche, lo stato del server è sempre waiting.

2.2.3 Priority Exchange Server

Oltre al diagramma temporale, devo costruire un diagramma per la capacità $Cs(p_s)$ e uno per tutti i processi con priorità minore.

1. La capacità può essere accumulata al livello di priorità di P_s anche al livello dei processi P_j meno prioritari $p(P_j) < p(P_s)$.

2. La capacità disponibile a un qualunque livello di priorità è conservata durante l'esecuzione di un processo di priorità maggiore o uguale.

3. La capacità viene conservata in assenza di richieste aperiodiche pendenti.

4. La capacità disponibile al massimo livello di priorità è consumata durante il servizio di richieste aperiodiche

5. La capacità disponibile al massimo livello di priorità è consumata se il sistema è tutto idle. Se $Cs(max) = 0$ ed è tutto idle, allora consumo le Cs a priorità minore.

6. In assenza di richieste aperiodiche, la capacità disponibile al massimo livello di priorità è trasferita al livello di priorità dei processi di priorità minore pronti per l'esecuzione. item In presenza di richieste aperiodiche

P_s assume la priorità del processo a priorità maggiore con capacità positiva. In questo caso P_s ha la precedenza rispetto a tale processo, quidni se tale processo ha bassa priorità, può verificarsi inversione di priorità con i processi ad alta priorità

La schedulabilità è garantita se vale:

$$U_p \leq (N+1)(2^{1/(N+1)} - 1) - U_s \tag{26}$$

oppure se PES ha priorità massima deve valere

$$U_s \leq \frac{2}{(1 + U_p/N)^N} - 1 \tag{27}$$

2.2.4 Sporadic Server

1. La capacità viene conservata in assenza di richieste aperiodiche pendenti.

2. La capacità è consumata durante il servizio di richieste aperiodiche

3. In assenza di processi periodici pronti la capacità è ripristinata al valore massimo.

4. La capacità è reintegrata dopo essere stata consumata e nella misura in cui è stata consumata. L'istante RT e l'entità di reintegro RA sono calcolati come segue a partire dall'istante t_A in cui si verifica la condizione $C_s > 0$ e P_s attivo (P_s è attivo se il processo in esecuzione ha priorità maggiore o uguale a quella di P_s) e t_D il successivo istante in cui una delle due condizioni non è rispettata:

$$RA = cap\_consumata\_in[t_A, t_D] \tag{28}$$
$$RT = \max\{t_A + T_s, t_D\} \tag{29}$$

Nella versione base a volte SS rende l'insieme di processi non schedulabile per via dei rimasugli di capacità non utilizzata: i chunck servono per tener conto di questo fatto.
Quando il carico delle richieste aperiodiche non comporta il consumo di tutta la capacità C_s, vengono creati dei *chunks* di capacità non utilizzata e consumata da gestire separatamente, tenendo traccia per ognuna di esse del corrispondente tempo di reintegro RT. Le regole di reintegro dei singoli

chunks, di volta in volta utilizzati per il servizio delle richieste aperiodiche in ordine di RT crescente sono generalizzate come segue:

$$t_E = \max\{RT, t_A\} \tag{30}$$
$$RA = cap\_consumata\_in[t_E, t_D] \tag{31}$$
$$RT = \max\{t_E + T_s, t_D\} \tag{32}$$

La schedulabilità è garantita se vale:

$$U_s \leq (N+1)(2^{1/(N+1)} - 1) - U_p \tag{33}$$

oppure se SS ha priorità massima deve valere

$$U_s \leq \frac{2}{(1 + U_p/N)^N} - 1 \tag{34}$$

2.3 Servizio tramite server a priorità dinamica

2.3.1 Constant Utilization Server

1. P_s attende le richieste aperiodiche; appena arriva una richiesta all'istante t_{Ra} vengono calcolati

   $$c_S = C_{Ra}, \qquad d_S = t_{Ra} + c_S/U_S \tag{35}$$

   P_S viene eseguito in accordo alla priorità dinamica correlata alla deadline d_S.

2. Eventuali richieste pervenute prima di d_S vengono accodate.

3. All'istante d_S, se ci sono richieste pendenti, oppure al sopraggiungere di una nuova richiesta dopo d_S, viene identificato il tempo di servizio C_{Ra} della richiesta in testa alla coda e si pone:

   $$c_S = C_{Ra}, \qquad d_S = \max\{d_S, t_{Ra}\} + c_S/U_S \tag{36}$$

2.3.2 Total Bandwith Server

1. P_s attende le richieste aperiodiche; appena arriva una richiesta all'istante t_{Ra} vengono calcolati

   $$c_S = C_{Ra}, \qquad d_S = t_{Ra} + c_S/U_S \tag{37}$$

   P_S viene eseguito in accordo alla priorità dinamica correlata alla deadline d_S.

2. Eventuali richieste pervenute prima del completamento del servizio vengono accodate (questo significa che all'interno di un periodo possono essere soddisfatte più richieste).

3. Appena terminato il servizio precedente, se ci sono richieste pendenti, oppure al sopraggiungere di una nuova richiesta, viene identificato il tempo di servizio C_{Ra} della richiesta in testa alla coda e si pone:

$$c_S = C_{Ra}, \qquad d_S = \max\{d_S, t_{Ra}\} + c_S/U_S \tag{38}$$

3 Schedulazione di processi con risorse condivise

3.1 Non-Preemptive Critical Section Protocol

3.2 Priority Inheritance Protocol

3.3 Priority Ceiling Protocol

3.4 Immediate Priority Ceiling Protocol

3.5 Dynamic Priority Ceiling Protocol

3.6 Preemption Ceiling Protocol

3.7 Stack Resource Policy